Cartographic documents for web modeling and representation of indoor mapping with interactive environments

[Extended Abstract] *

Ben Trovato†
Institute for Clarity in Documentation
1932 Wallamaloo Lane
Wallamaloo, New Zealand
trovato@corporation.com

G.K.M. Tobin‡
Institute for Clarity in Documentation
P.O. Box 1212
Dublin, Ohio 43017-6221
webmaster@marysville-ohio.com

Lars Thørväld§
The Thørväld Group
1 Thørväld Circle
Hekla, Iceland
larst@affiliation.org

Lawrence P. Leipuner
Brookhaven Laboratories
Brookhaven National Lab
P.O. Box 5000
lleipuner@researchlabs.org

Sean Fogarty
NASA Ames Research Center
Moffett Field
California 94035
fogartys@amesres.org

Charles Palmer
Palmer Research Laboratories
8600 Datapoint Drive
San Antonio, Texas 78229
cpalmer@prl.com

ABSTRACT

PUT THE ABSTRACT HERE

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Theory

Keywords

ACM proceedings, LaTeX, text tagging

1. INTRODUCITON

...

*A full version of this paper is available as *Author's Guide to Preparing ACM SIG Proceedings Using LaTeX2ε and BibTeX* at `www.acm.org/eaddress.htm`

†Dr. Trovato insisted his name be first.

‡The secretary disavows any knowledge of this author's actions.

§This author is the one who did all the really hard work.

The remainder of the paper is organized as follows. In Section II is provided an overview of the state of the art. Section III is devoted to describe the novel cartographic document proposed, while section IV introduces the underlying mathematical structure. Section V reports about the tools and instruments developed specifically for the web, focusing on software architecture, implemented algorithms and real applications. Section VI describes one use case of both document format and software tools. Finally Section VII proposes some conclusive remarks and future developments.

2. STATE OF THE ART

PUT THE STATE OF THE ART HERE

2.1 GeoJSON

GeoJSON is a format for encoding a variety of geographic data structures. GeoJSON supports the following geometry types: `Point`, `LineString`, `Polygon`, `MultiPoint`, `MultiLineString`, and `MultiPolygon`. Lists of geometries are represented by a `GeometryCollection`. Geometries with additional properties are `Feature` objects. And lists of features are represented by a `FeatureCollection`. GeoJSON%20spechttp://ge spec.html

GeoJSON is good for geographic mapping application, but not for indoor application. There is a GeoJSON variant suitable for indoor app.

2.2 Experiences on Indoor JSON

`IndoorJSON` is a GeoJSON variant used by indoor.io toolset to define indoor maps. IndoorJSON may consist of any number of Features and/or FeatureCollections. All Features are interpreted similarly regardless of their grouping into nested FeatureCollections. IndoorJSON supports all GeoJSON geometry types.

3. ADVANCES ON CARTOGRAPHICS DOCUMENT STANDARDS

HIJSON (**H**ierarchical **I**ndoor **JSON**) is a GeoJSON variant. A HIJSON document reveals at least three major enhancements above the actual state of the art in indoor cartographic documents:

1. Hierarchical structure
2. Metric local coordinate System
3. Semantic extensions

3.0.1 Hierarchical structure

Unlike other formats like GeoJSON or IndoorJSON, HIJSON organizes its elements in a hierarchical structure, where every element represent a potential container for other elements. This structure allows a clear and logical organization of the elements inside the structure, and at the same time make it possible to use a relative, local, metric coordinates system.

3.0.2 Metric local coordinate System

In GeoJSON all the positions are expressed in geographical coordinates (usually WGS84). Although this can be useful for outdoor geographical representations, it is not the best solution for indoor descriptions. In HIJSON all the coordinates are expressed in a relative system based on the hierarchical structure. The shape of all elements is described starting from origin, and then two vectors (translation and rotation) describe the position relative to the origin of the parent element. By this way it is possible to describe the position of a piece of furniture by specifying its distance from the origin of the room, that is obviously more convenient than describing its geographical coordinates. Another advantage is represented by the adoption of a metric reference. A recursive process that computes intermediate transformation matrixes can then produce a standard GeoJSON representation, that can be visualized on any standard viewer.

3.0.3 Semantic extensions

Every HIJSON Element has a property that describes its class. This information allows the adoption of semantic extensions by the software that manipulates the HIJSON data. In the Javascript library developed to manage HIJSON documents, different classes are instantiated to represent HIJSON Nodes, which acts differently by the adoption of polymorphic methods. Ino rder to extend the possibilities in representation and interaction, it is sufficient to define new classes that reflects new categories of HIJSON Elements.

4. LAR: THE UNDERLYING MATHEMATICAL STRUCTURE

rules

A single HIJSON document is composed of different parts: - configuration: a JSON object containing parameters and settings useful for the building representation. In particular three points of the local reference system are mapped to three couples of geographical coordinates. This information allows the computation of the transformation matrix used to translate the local coordinates to global ones. - one or more data collections: each of these lists is given in the form of a GeoJSON FeatureCollection, containing a number of HIJSON Elements. Since HIJSON Elements adhere to the GeoJSON format, each collection can be accepted by a GeoJSON validator. HIJSON introduces some additional rules that allow the adoption of this format for indoor representation. Below is given a sample of HIJSON Element, with the description of the main differences from a standard GeoJSON Feature.

```
{
    "type": "Feature",
    "id": "room_0.1",
    "geometry":
    {
        "type": "Polygon",
        "coordinates":
        [
           [ [0, 0], [11, 0], [11, 19], [0, 19] ]
        ]
    },
    "properties":
    {
        "class": "room",
        "parent": "level_0",
        "description": "Office of Mr. Smith",
        "tVector": [10, 20, 0],
        "rVector": [0, 0, 90]
    }
}
```

The first additional requisite above the GeoJSON format rules is the necessity of a unique ID, necessary for the referencing by possible child elements. The Geometry types allowed are `Point`, `LineString` and `Polygon`. Each geometry type is used to represent particular categories of elements (e.g. Polygons for levels and rooms, LineString for walls and doors, Point for furniture, etc.). The geometry coordinates are expressed in meters, and for convention starting at the bottom-left of the element. Unlike GeoJSON, where all the properties are optional, in HIJSON some attributes are mandatories: - `class`: represent the element category, used to instantiate the appropriate semantic class; - `parent`: contains parent's id of the nodes. The reason of the unique id depends on this property. The HITREE is created on the base of parent property; - `tVector` and `rVector`: represent the translation and rotation relative to the parent element. The measure unit for translation is meter and for rotation is grades.

The definition of other properties is mandatory on the base of the class of the element: For example the classes that defines internal or external walls require a `connections` array, containing the IDs of the adiacent areas. This information is used by the connector children of the elmenet, like doors, to identify the areas linked together. These connector elements, like doors, are identified bay a boolean `connector` property set to true.

Optional fields can be added to improve the precision of the representation. Given the nature of the GeoJSON format from which HIJSON derives, the elements are represented by their 2D shape, like on a planimetry. To assign a value to the height of the object, intended as third dimension, the property `height` can be used.

A `description` property can provide additional information about the element.

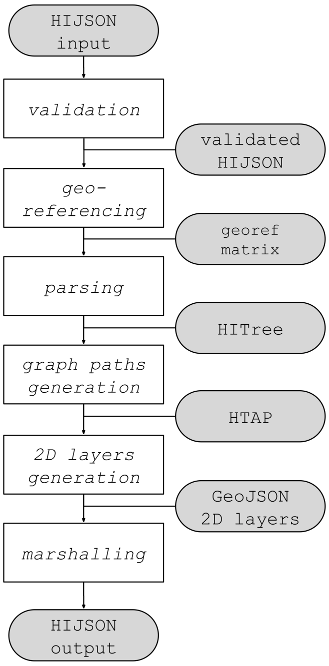

Figure 1: HIJSON processing pipeline

Additional optional fields can be freely added, to enrich and extend the expressvity of the representation.

5. IMPLEMENTED WEB TOOLS

A set of web based instruments has been developed allowing to deal with the HIJSON document previously described. Tools are written in *JavaScript* language, using *Node.js* and in particular *Express.js* as backend framework, and exploiting the power of WebSocket protocol through the *Socket.io* library.

5.1 Architecture

The overall architecture of the tools are, as for the vast majority of the web based application, inherently *client/server*.

5.1.1 HIJSON processing pipeline

Each time a new HIJSON document is submitted to the server, it is passed through the **HIJSON processing pipeline**, where it is subjected to a sequence of preliminary transformations.

The application of the transformation pipeline has a double aim. The first one consists in generating the graph of valid paths between all the interesting HIJSON elements. The second aim is the generation of one *GeoJSON* document for each story of the building described by the HIJSON document. In this way any connected client can be provided with a bidimensional plant for each level of the building that it can visualize through any compliant GeoJSON viewer.

HIJSON processing pipeline (as pictured in figure ...) is composed by 6 elaboration stages. In the following are detailed operations excecuted by each stage, which are, in the order: *validation*, *georeferencing*, *parsing*, *graph paths generation*, *2D layers generation*, *marshalling*.

1. [**validation**] - The first one is a validation stage. In order to begin with the effective transformations the input HIJSON document must be compliant with the rules defined in (AGGIUNGERE REF TO PARAGRAFO REGOLE DI VALIDITA'). In the case the validation stage fails, processing aborts and do not continue to following stages. If the stage success, the output for the next stage is a validated HIJSON. 2. [**georeferencing**] - 3. [**parsing**] - The parsing stage, takes the validated HIJSON as its input, that as illustrated before can be thought of as a list of HIJSON Elments, parses them and produce an istance of HITREE, which is an object in memory representing the tree hierarchical structure of the building described into the HIJSON. 4. [**graph paths generation**] - The fourth stage is in charge of THE generatio of the graph paths. This aim is accomplished according to the algorithm described in (AGGIUNGERE RIFERIMENTO A #### Automatic generation of valid paths). The graph paths will be useful afterwards to coumpute valid paths from couple of point of interest on the graph. Once the graph paths has been computed, the input HITREE is augmented with paths information, becoming what has been called an HTAP (HITREE Augmented with Paths). Augmentation always takes place as leaf nodes added as children of a specific (e.g. "room") level. 5. [**2D layers generation**] - The fifth state is the generation of geoJSON layer. For each level, the system generates one geoJSON layer that will be use for the creation of 2D map. Each layer contains the children of 'level' node in the HITREE. Every class contains a boolean value that is use to choose which class will be a part of geoJSON layer. Every element has a geographical coordinates calculated by the transformation matrix with regard to the local coordinates of the HIJSON element. 6. [**marshalling**] -

5.1.2 Client

When a client connects to the server, it receives the HIJSON input files, the ready-to-use GeoJSON layers and the weighted adjacency matrix of the grapth paths. A very short pipeline of processing is performed by the client, composed by: 1. [**Parsing**] - Like for the Server-side processing, the HIJSON Elements in the input files are processed and tranformed in HIJSON Nodes, linked together in a hierarchical tree structure. 2. [**3D Model generation**] - Unlike HIJSON Elements (that are simple Javascript objects), HIJSON Nodes are instances of specific classes, representing a particular category of element in the building. Through a polymorphic method, each node generates a Three.js 3D model of its entity, that is used to assemble a complete 3D Model of the building. The similarities in HIJSON hierarchical structure and Three.js scene graph allow this process to be performed with little effort.

5.2 Algorithmics: automatic generation of valid paths

(IN QUESTA SEZIONE SI POSSONO AGGIUNGERE EVENTUALI APPROFONDIMENTI DI ALTRI STADI DELLA PIPELINE)

The fourth stage of the processing pipeline is responsible for the generation of a graph of valid paths through the entire model represented by the intput HIJSON document. The graph generated according to the algorithm described in the following, although not optimal, ensures a complete coverage of the surface while limiting the numebr of generated nodes. Resulting graph is weigthed on the edges with nodes distances and each node represents alternatively:

a. standard path node, i.e. a junction node or possibly an endpoint of a path;
b. connection node, used as subproblem composing element in the divide et impera approch adopted (as described below).
c. element nodes ie. HIJSON Element (whose HIJSON Class explicitly grants his presence in the graph), typically an endpoint of a path;

Such a graph allows for directions calculations between any two given nodes. Directions are actually computed clientside applying the Dijkstra's shortes route algorithm on the graph.

5.2.1 Graph paths generation

Taking advantage of the hierarchical structure of the HIJSON document, and according to the divide et impera approach, the problem of the graph paths generation is splitted in several sub-problems which consist in the computation of the sub-graphs relative to each room, or more generally ambience. The sub-graphs are then linked together through the connection nodes (which in most cases represents doors). The resolution of each sub-problem (as depicted in figure METTERE RIFERIMENTO ALLA FIGURA), is composed by 4 phases: 1. Computation of the walkable area of the ambience: this task is accomplished subtracting area of the possibly encumbrances to the area of the ambience; the result is tipically a surface with holes; 2. Triangulation of the walkable area: the computed surface is triangulated taking into account the presence of holes; 3. Identification of graph nodes: for each triangle side completely internal to the area, its midpoint is selected as standard path node; 4. Junction of nodes: nodes relative to the same triangle are then linked together; both element nodes and connection nodes (i.e. doors) are linked to the nearest node in the ambience (i.e. room).

5.3 HIJSON Class definition

Every HIJSON Elment is assigned to a specific class, i.e. category, through the "class" property. This information can be used by an application, in this case the Web Toolkit, to instantiate specific classes with differentt behaviours. These HIJSON Classes, when instantiated, produce an HIJSON Node, that is the building block of the HITREE.

The HIJSON Web Toolkit provides a number of HIJSON Classes that are used to characterize the representation and behaviour of some basic indoor elements. In order to extend the possibilities in terms of differentiation it is possible to add new classes that reflect new categories of elements. Below is described the structure of a HIJSON Class.

Each class extends a base Feature Class, that provides basic and common properties across all the elements. In particular the basic constructor copies all the the properties from the HIJSON Element to the just created HIJSON Node. In every sub-class, some methods and properties are defined to specify the behaviour of the element, for example: - style: this property is used by the client to apply specific visualization rules in the 2D Leaflet map. - render: this method returns a Object3D used for rendering of 3D model. - getInfo: a method that return a React Component, which populates the DOM with a specific UI for retrieving information and interact with the object. - visible_2D: a boolean value that indicates the - in_graph: a boolean value that indicates if the HIJSON element will be represent with a node in a graph.

images/extended-classes-eps-converted-to.pdf

Figure 3: Semantic classes

If a method or property is non overrided by the sub-class, the defaults defined in the Feature super-class are adopted (default 2D style, empty 3D Object, static informations etc.).

The definition of classes permits to extend the software in function of user's necessity.

5.3.1 Example of use

The HIJSON Web Toolkit takes advantage of the semantic classes boh in the server-side processing and the client-side visualization. During the server-side processing pipeline the information retreived from the "visible_2D" and "in_graph" properties are used to evaluate the inclusion of the element in the GeoJSON layers or in the graph of paths. In client-side there are essentially two representation: 2D and 3D. In 2D map will be a representation by calling a polymorphism method that respect the style defined in the class. The style is a simple object that contains the graphical properties of the class. For 3D model, the "render" method returns a Object3D in according three.js library. If the method is not defined in the class, then will be used a "render" method of the superclass by default: returns an empty object3D. For the IoT monitoring, in every class is defined a method, 'getInfo'. It returns a React component that contains a general information of the object(by calling to superclass method) and custom element of the object. By default, the superclass method return general information, like the position and name of the object. # Conclusions We presented HIJSON a GeoJSON extension for indoor mapping TRA GLI SVILUPPI FUTURI: - GENERAZIONE GRAFICA IN AMBIENTE CAD DEL DOCUMENTO HIJSON

6. CONCLUSIONS

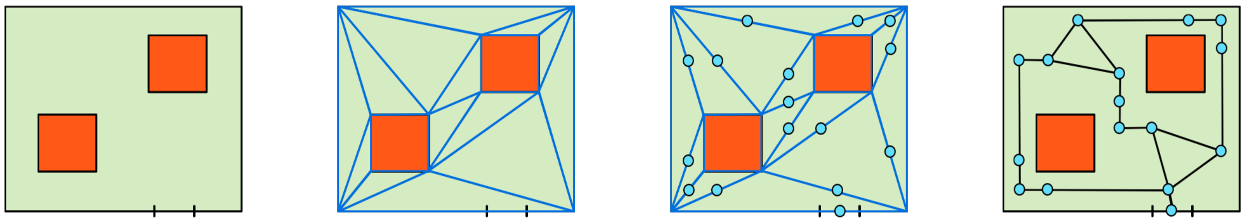

Figure 2: Graph paths generation. (a) detection of obstacles; (b) triangulation of walkable area; (c) identification of graph nodes area; (d) junction of nodes.

We presented HIJSON a GeoJSON extension for indoor mapping TRA GLI SVILUPPI FUTURI: - GENERAZIONE GRAFICA IN AMBIENTE CAD DEL DOCUMENTO HIJSON

7. ACKNOWLEDGMENTS

This section is optional; it is a location for you to acknowledge grants, funding, editing assistance and what have you. In the present case, for example, the authors would like to thank Gerald Murray of ACM for his help in codifying this *Author's Guide* and the **.cls** and **.tex** files that it describes.

APPENDIX

A. HEADINGS IN APPENDICES

The rules about hierarchical headings discussed above for the body of the article are different in the appendices. In the **appendix** environment, the command **section** is used to indicate the start of each Appendix, with alphabetic order designation (i.e. the first is A, the second B, etc.) and a title (if you include one). So, if you need hierarchical structure *within* an Appendix, start with **subsection** as the highest level. Here is an outline of the body of this document in Appendix-appropriate form:

A.1 Introduction

A.2 The Body of the Paper

A.2.1 Type Changes and Special Characters

A.2.2 Math Equations

Inline (In-text) Equations.

Display Equations.

A.2.3 Citations

A.2.4 Tables

A.2.5 Figures

A.2.6 Theorem-like Constructs

A Caveat for the TEX Expert

A.3 Conclusions

A.4 Acknowledgments

A.5 Additional Authors

This section is inserted by LATEX; you do not insert it. You just add the names and information in the `\additionalauthors` command at the start of the document.

A.6 References

Generated by bibtex from your .bib file. Run latex, then bibtex, then latex twice (to resolve references) to create the .bbl file. Insert that .bbl file into the .tex source file and comment out the command `\thebibliography`.

B. MORE HELP FOR THE HARDY

The sig-alternate.cls file itself is chock-full of succinct and helpful comments. If you consider yourself a moderately experienced to expert user of LATEX, you may find reading it useful but please remember not to change it.